

INHIBITION FEATURE TO SYSTEM FOR PREDICTION AND  
CONTROL OF POWER CONSUMPTION IN DIGITAL SYSTEMS

by

Kevin D. Kissell

Assignee: MIPS Technologies Inc.  
1225 Charleston Road  
Mountain View, CA 94043-1353

Address correspondence to:

James W. Huffman  
1832 N. Cascade Ave.  
Colorado Springs, CO 80907  
719.475.7103  
719.623.0141 (fax)  
jim@huffmanlaw.net

INHIBITION FEATURE TO SYSTEM FOR PREDICTION AND  
CONTROL OF POWER CONSUMPTION IN DIGITAL SYSTEMS

by

Kevin D. Kissell

---

CROSS REFERENCE TO RELATED APPLICATION(S)

[0001] This application is a continuation-in-part of co-pending U.S. Patent Application Serial No. 09/665099 (Docket MIPS:0103.00US), filed September 20, 2000, entitled "SYSTEM FOR PREDICTION AND CONTROL OF POWER CONSUMPTION IN DIGITAL SYSTEMS", commonly assigned to MIPS Technologies, Inc.

BACKGROUND OF THE INVENTION

Field of the Invention

[0002] This invention relates in general to the field of instruction processing in computer systems, and more particularly to an apparatus and method in a CPU for predicting power consumption for executing instructions and counterbalancing or randomizing total power consumption for security purposes by turning on or shutting down selected subsystems.

## Description of the Related Art

[0003] Early computers processed instructions, one at a time, with the instructions appearing on a data/address bus. The data/address bus was electrically visible to a test engineer such that probes could be directly attached to the bus, and the bus could be monitored to determine what instructions were executing. In fact, in embedded system design, it was common for a developer to utilize a test device, such as a state analyzer, to help test and debug his program.

[0004] For example, a developer would connect a state analyzer to the data/address bus of his computer, load his program into the memory of the computer, and then reset the computer so that it would begin processing his instructions. The state analyzer would record the instructions being executed on a clock-by-clock basis. The developer could examine the instructions recorded by the state analyzer, and could determine whether his program was executing as planned. In a similar manner, someone wanting to discover what program was operating on a computer could connect a state analyzer to the data/address bus, and could capture the program as it executed (i.e. reverse engineer the code).

[0005] With the advent of the microprocessor, testing of program code, and reverse engineering of program code, became more difficult. Early microprocessors utilized address and data buses to obtain their instructions and data from external memory such as ROM's and RAM's. Thus, a state analyzer could be coupled to the address and data buses of a microprocessor to capture the values appearing on the buses on a clock-by-clock basis. Software programs, called disassemblers, were then used to interpret the values appearing on the address and data buses, and convert them into the actual instructions that were being executed by the microprocessor. Thus, state analyzers, along with disassemblers, were used both by software developers for debugging, and by those interested in discovering what someone else had programmed.

[0006] As the complexity of microprocessors increased, so did the difficulty of testing (or reverse engineering) the instructions they were executing. For example, modern processors, although they have external data and address buses, cannot successfully be probed to determine what instructions are executing. This is because the values appearing on the address and data buses seldom relate to what the processor is doing at any one time.

[0007] More specifically, within a modern microprocessor are large areas of memory (or cache) for both data and instructions. When a processor needs data or instructions that are not already resident within the processor, the processor will request the data/instructions from external memory. It is common, however, for blocks of data/instructions, in addition to those particularly requested, to be transferred back to the processor. So, examination of the external data/address buses during the transfer does not provide any direct correlation to the instructions that are executing. In addition, as is most often the case, the data and instructions that are executing are already resident within internal cache. So, as they execute, there is no activity on the external address/data buses that provide any indication as to what is being executed.

[0008] Furthermore, most modern processors are capable of executing multiple instructions at the same time, using a technology referred to as "pipelining". Although pipelining is beyond the scope of the present discussion, it suffices to say that pipelining in a microprocessor is similar to an assembly line, where an instruction proceeds thru several stages within a processor, each stage

performing a certain operation. The instruction is preceded and followed by other instructions, in preceding and following stages in the pipeline. Thus, at any one time, a pipelined microprocessor is executing as many different instructions as there are stages in the pipeline (e.g., a 5 stage pipeline may be executing 5 different instructions).

[0009] So, since there is no longer a direct correlation between what appears on the data/address buses of a processor, and the instructions it is executing, and because multiple instructions are typically executing simultaneously, other methodologies have been developed for both software test, and reverse engineering. For software test, a common approach is the use of a simulator. The simulator utilizes a software model of the internal architecture of a processor (such as a VHDL model) to execute a particular program. A trace output from the simulator provides an indication to a developer of how his program should function within a fabricated processor. The simulator is a great tool for the developer, because s/he has access to the internal model of the processor.

[0010] The reverse engineer, however, has no access to the internal model of a processor that s/he is trying to

decode. Such an internal model is highly proprietary, and closely guarded by the developer. But the reverse engineer has not been thwarted. Rather, a sophisticated science, referred to as "differential power analysis" (DPA), has been developed to speculate or determine the instruction flow within a processor.

[0011] In general, certain estimates can be made about the size and thus the power requirements of particular functional blocks (cache, ALU, Register, etc.) of a modern processor. Using these estimates, the total power consumed by a microprocessor can be monitored on a clock-by-clock basis, while the microprocessor is executing instructions. Analysis of the power consumed by the microprocessor, in conjunction with speculated power requirements associated with each of the instructions in the processor's instruction set, can often allow a reverse engineer to accurately speculate or determine the instructions being executed by a processor.

[0012] In many cases, such reverse engineering can have devastating consequences. For example, one modern application of a microprocessor is within a smart card. The smart card is much like a credit card. However, it contains a microprocessor that stores confidential

information such as a person's bank account number, balance, medical history, social security number, etc. Thus, the developer of a smart card is very interested in preventing the information within the processor from being discovered. Differential power analysis applied to reverse engineer the processor can allow such confidential to be discovered, and utilized, to the detriment of the cardholder.

[0013] In view of the above, what is needed is an apparatus that may be used within a microprocessor, to prevent the reverse engineering of the program that is executing on the processor.

[0014] More specifically, what is needed is a power management apparatus that can disassociate the power that is consumed by a microprocessor, during instruction execution, from the actual power requirements of those instructions. Such an apparatus, in disassociating the power consumed by the microprocessor, from the instructions it is executing, would thwart a reverse engineer's attempt to discover confidential information using differential power analysis.

[0015] Moreover, what is needed is a power management apparatus that turns on or shuts down certain subsystems



within the microprocessor, or subsystems that are external to but associated with the microprocessor (i.e., such as a system on a chip), for the purpose of disassociating the instantaneous power drain of the microprocessor from the instructions it is executing.

#### SUMMARY

[0016] For the foregoing reasons, there is a need for an apparatus and method that disassociates the power consumption of a microprocessor based system from the instructions that it is executing.

[0017] To address the above-detailed deficiencies, it is an object of the present invention to provide an apparatus within a processing system having multiple subsystems, for disassociating the power consumed by the processing system with instructions it is executing. The apparatus includes a power prediction system and a subsystem inhibition control. The power prediction system provides a predictor of estimated power that will be consumed during execution of instructions. The subsystem inhibition control receives the predictor, and turns on/off selected ones of the subsystems based on the value of said predictor. By turning on/off ones of the selected subsystems, the subsystem inhibition control causes a total power

consumption of the processing system to be disassociated with the instructions it is executing.

[0018] In another aspect, the present invention provides a security system for a microprocessor having multiple subsystems to insure that the power consumption of the microprocessor is not indicative of the instructions that it is executing. The security system includes a power correlator and subsystem inhibition control. The power correlator estimates the power that the microprocessor will consume during execution of instructions. The subsystem inhibition control, utilizes the estimated power, and selectively turns on/off ones of the multiple subsystems within the microprocessor during execution of the instructions. The ones of the multiple subsystems that are turned on/off are selected so as to disassociate the total power consumed by the microprocessor from the instructions it is executing.

[0019] In yet another aspect, the present invention provides an apparatus within a processing system having multiple subsystems for randomizing the total power consumed within the processing system. The apparatus includes a random value generator, and subsystem inhibition control. The random value generator generates a random

value as instructions are executed by the processing system. The subsystem inhibition control turns on/off selected ones of the multiple subsystems in response to the generated random value. By turning on/off selected ones of the subsystems, the subsystem inhibition control disassociates instructions that are executed from power consumed by the processing system during their execution.

[0020] In a further aspect, the present invention provides a computer program product for use with a computing system. The computer program product includes a computer usable medium, having computer readable program code embodied in said medium, for causing a processing system having multiple subsystems to be described. The computer readable program code includes: first program code for providing a power prediction system for providing a predictor of estimated power that will be consumed during execution of instructions by the processing system; and second program code for providing a subsystem inhibition control, coupled to said power prediction system, for receiving said predictor, and for turning on/off selected ones of the subsystems based on the value of said predictor.

[0021] Yet a further aspect of the invention provides a method for disassociating the power that is consumed by a microprocessor having multiple subsystems, from the instructions that it is executing. The method includes: estimating the power that will be consumed by each instruction in the microprocessor; and selectively turning on/off ones of the multiple subsystems such that the total power consumed by the microprocessor is unrelated to the instructions that are executing.

[0022] In a further aspect, the present invention provides a computer data signal embodied in a transmission medium including computer-readable first program code for providing a power correlator that estimates the power that will be consumed as an instruction executes on a microprocessor having multiple subsystems; and computer-readable second program code for providing subsystem inhibition control, for utilizing the estimated power, and for selectively turning on/off ones of the multiple subsystems within the microprocessor during execution of the instructions.

#### BRIEF DESCRIPTION OF THE DRAWINGS

[0023] These and other objects, features, and advantages of the present invention will become better understood with

regard to the following description, and accompanying drawings where:

[0024] FIGURE 1 is a prior art diagram illustrating the use of differential power analysis to determine instruction flow in a microprocessor.

[0025] FIGURE 2 is a prior art block diagram illustrating power variation per clock cycle of a sequence of executing instructions in a pipeline microprocessor.

[0026] FIGURE 3 is a block diagram of a predicted power profile table according to the present invention that predicts power consumption, per pipeline stage.

[0027] FIGURE 4 is a block diagram of the predictive power correlation according to the present invention, within a pipeline microprocessor.

[0028] FIGURE 5 is a diagram illustrating the power consumption of a pipeline microprocessor incorporating the power prediction and counterweight according to the present invention.

[0029] FIGURE 6 is a block diagram illustrating per stage power prediction logic within a pipeline processor according to the present invention.

[0030] FIGURE 7 is a block diagram of a random power counterweight circuit within a pipeline microprocessor according to the present invention.

[0031] FIGURE 8 is a graph illustrating the random power counterweight circuit of Figure 7.

[0032] FIGURE 9 is a block diagram of a microprocessor incorporating subsystem inhibition control according to the present invention.

[0033] FIGURE 10 is a graph illustrating the effect of the subsystem inhibition control on the instantaneous power drain of a microprocessor to maintain a constant power drain according to the present invention.

[0034] FIGURE 11 is a graph illustrating the effect of the subsystem inhibition control on the instantaneous power drain of a microprocessor to provide a random power drain according to the present invention.

#### DETAILED DESCRIPTION

[0035] Referring to Figure 1, a block diagram of a microprocessor based system 100 is shown to illustrate a prior art method of determining what instructions are executing on a microprocessor. The method utilizes a

technology referred to as differential power analysis. More specifically, a microprocessor 102 is shown attached to a differential power analyzer 104 via current probes 106. In most instances, the differential power analyzer 104 is a current monitor that is capable of measuring current drain on selected pins of the microprocessor 102, on a continuous basis. Since the voltage is constant, power consumed by the microprocessor 102 may be determined by measuring the current drawn on each of its power supply pins. A graph 108 is shown illustrating the cumulative current (measured along the y-axis) drawn by the microprocessor 102 during consecutive clock cycles (measured along the x-axis). Also shown are two scientists 110 examining the chart 108 to try to understand what the microprocessor is doing during each clock cycle.

[0036] For example, it is possible to characterize functional blocks 103 within a microprocessor 102, such as floating point units, integer Arithmetic Logic Units (ALU's), cache memory systems, etc., in terms of how much power is required for the units to perform a task during a clock cycle. Once such a characterization is done (or estimated), measurement of current consumed by the microprocessor 102 can allow fairly accurate speculation,

if not determination, of exactly which instructions are executing on the microprocessor 102, even though the scientists 110 do not have access to programmed instructions, as written, or to the internal buses of the microprocessor 102. And, by determining what the instructions are that are executing on the microprocessor 102, security features of the microprocessor based system 100, may be broken.

[0037] Referring now to Figure 2, a prior art block diagram 200 is shown to particularly illustrate the effect of different instructions, or instruction types, on the power consumption of a pipelined microprocessor. More specifically, a 5-stage microprocessor 200 is shown. The stages include: Fetch 202, Register 204, ALU 206, Memory 208 and Write 210. The Fetch stage 202 utilizes a program counter (not shown) to fetch instructions to be executed by the processor 200. Once fetched, the instructions are provided to the Register stage 204. The Register stage 204 decodes the instructions, and if necessary, retrieves the contents of specified operands and provides them to the ALU stage 206. The ALU stage 206 either performs arithmetic on the operands, or calculates addresses, for load/store instructions. Instruction flow then proceeds to the Memory



stage 208. The memory stage 208 is typically associated with cache access for load/store operations. Instruction flow then proceeds to the Write stage 210 where the result of ALU operations, or data retrieved from the cache, are written back to the register file or memory (not shown).

[0038] For exemplary purposes, a table 220 is provided to illustrate the progression of eight instructions ( $I_1$  -  $I_8$ ) through the pipeline stages 204-210 of the processor 200. One skilled in the art will appreciate that the current values associated with each of the instructions is illustrative only, and no meaning should be attached to the specific values shown.

[0039] During a first clock cycle, instruction  $I_1$  is in the W stage 210, instruction  $I_2$  is in the M stage 208, instruction  $I_3$  is in the A stage 206, and instruction  $I_4$  is in the R stage 204. As shown, each of the instructions  $I_1$  thru  $I_4$  consumes different amounts of power (5 - 30) depending on what function they are performing, and depending on which pipeline stage they are in. In this illustration, their combined current consumption is 65.

[0040] During a second clock cycle, instruction  $I_1$  falls out of the pipeline, instructions  $I_2$ - $I_4$  proceed to the next pipeline stage, and instruction  $I_5$  enters the R stage 204.

At this point, two things should be understood. First, instructions typically proceed sequentially from the R stage 204, thru to the W stage 210. Second, the amount of current consumed by a particular instruction is not constant through each of the stages. Rather, the current consumed by a particular instruction will often vary from stage to stage, with the amount in each stage determined by the type of instruction executing. In the second clock cycle, the combined current consumption is shown to be 70.

[0041] During a third clock cycle, instruction  $I_2$  falls out of the pipeline, instructions  $I_3$ - $I_5$  proceed to the next pipeline stage, and instruction  $I_6$  enters the R stage 204. In the third clock cycle, the combined current consumption is shown to be 45.

[0042] During a fourth clock cycle, instruction  $I_3$  falls out of the pipeline, instructions  $I_4$ - $I_6$  proceed to the next pipeline stage, and instruction  $I_7$  enters the R stage 204. In the fourth clock cycle, the combined current consumption is shown to be 60.

[0043] During a fifth clock cycle, instruction  $I_4$  falls out of the pipeline, instructions  $I_5$ - $I_7$  proceed to the next pipeline stage, and instruction  $I_8$  enters the R stage 204.

In the fifth clock cycle, the combined current consumption is shown to be 80.

[0044] The above is particularly illustrated by reference to chart 230 that graphically depicts the total current consumption of the processor 200 during clock cycles 1-5. It is information such as that shown within the chart 230 that is often used to break the security of a microprocessor 200, by determining what instructions are executing on it, on a clock-by-clock basis.

[0045] Referring now to Figure 3, a diagram 300 is shown of a power profile table 308 within a decoder 302 of a modern microprocessor. More specifically, the decoder 302 includes an instruction register 304 for receiving and temporarily storing instructions from a fetcher. The instruction register 304 is attached to an instruction decoder 306. The decoder 306 decodes the instructions in the register 304, and provides them to an instruction buffer 310.

[0046] In addition, as the instruction decoder 306 decodes instructions, it provides a signal to the power profile table 308 indicating the instruction that is being decoded. That is, it indicates either the type of instruction (Load, ALU, JMP, etc.), or alternatively, the

actual instruction within the instruction set of the microprocessor, which is being decoded. In one embodiment, the signal acts as an index into the power profile table 308 to select one of  $n$  different power profile entries in the table.

[0047] Each of the entries in the power profile table 308 contains an estimated power value, corresponding to a particular pipeline stage in the microprocessor. For example, the power profile entry associated with instruction  $I_4$  indicates that this instruction has a power value of 10 within the F stage, 20 within the R stage, 10 within the A stage, 20 within the M stage, and 10 within the W stage. As an entry in the power profile table 308 is indexed by the instruction decoder 306, the estimated power values are provided to a power profile register 312. In one embodiment, the power profile register 312 contains temporary storage for each of the power profile values for a particular instruction, along with an instruction indicator to designate which of the instructions the power profile entry 308 is associated with. One skilled in the art will appreciate that the values contained in a power profile entry may be determined heuristically by the manufacturer of a microprocessor prior to final

fabrication. In addition, it should be appreciated that the power profile values will vary in different processor designs, and with different technologies used to manufacture processors.

[0048] Referring now to Figure 4, a block diagram of a microprocessor based system 400 is shown illustrating a pipeline microprocessor, similar to that described above with reference to Figure 2, including however, the novel power counterweight system of the present invention. The microprocessor based system 400 includes a microprocessor having five pipeline stages: Fetch 402, Register 404, ALU 406, Memory 408, and Write 410. In one embodiment, the power profile table 308 as shown in Figure 3 is contained within the Register stage 404. As an instruction is decoded within the Register stage 404, it is provided to an instruction register 412. In addition, the decoder indexes into the power profile table 308, causing a power profile entry corresponding to the instruction to be stored into a power profile buffer 414. Then, as the instruction proceeds to each stage in the pipeline, the power profile entry associated with the instruction also proceeds down the pipeline, being stored in power profile buffers 414 located in each stage of the pipeline.

[0049] Coupled to each of the power profile buffers 414 are power predictors 416. The power predictors 416 extract the particular power profile value within the power profile buffer 414 that is appropriate for its pipeline stage. For example, the predictor 416 in the R stage will extract the R power profile value. The predictor 416 in the A stage will extract the A power profile value, etc.

[0050] Thus, for the microprocessor based system 400, at any one time, four distinct power profile values will be extracted by the power predictors 416, corresponding to four distinct instructions executing in the R, A, M and W stages of the pipeline. All of these power profile values are provided to a power correlator 418 for arithmetic combining. In one embodiment, the arithmetic combining is a simple summation of the values. In an alternate embodiment, the arithmetic combining allows predefined weighting to be applied to particular stages and/or allows minimum and maximum thresholds for each of the stages to be met, or substituted, for the values provided by the predictors 416. After the power correlator 418 performs the arithmetic combining, the combined power estimate is provided to a power counterweight 420.

[0051] In an alternate embodiment, rather than providing a power profile table 308 within the instruction decoder 302, a portion of the power profile table 308 may be placed within each power predictor 416 that is applicable for the pipeline stage within which the power predictor 416 resides. That is, power profiles for the R stage 404 may be placed within the power predictor 416 associated with the R stage, power profiles for the A stage 406 may be placed within the power predictor 416 associated with the A stage, and so on. Then, as each instruction proceeds thru the pipeline, the power predictor 416 for each pipeline stage indexes into its power profile table 308 and extracts a power profile for the instruction executing in its stage.

[0052] The power counterweight 420 receives the combined power estimate from the power correlator 418 and produces a counterweight current 422, the purpose of which is to make the total power consumed by the processor based system 400 to be invariant. That is, given a maximum power consumed by a processor based system 400, whenever the combined power estimate is less than the maximum power, the power counterweight 420 produces a counterweight current 422 that is equal to the maximum power minus the combined power estimate. This can be written as:

Counterweight Current = Max Power - Estimated Power

[0053] The maximum power threshold is provided via a maximum power threshold register 460 to the power counterweight 420. The contents of the power threshold register 460 may either be set from a pre-coded default 462, or alternatively, may be established at run-time through software configuration 464.

[0054] The power counterweight 420 is also coupled to a subsystem A 490 via an inhibit/burn signal line 492. Thus, as determined by the power counterweight 420, if additional power consumption may be met by turning on (burn) subsystem A 490 within the processor based system 400, the power counterweight indicates such to subsystem A 490 via the inhibit/burn signal line 492. This can be either in place of a counterweight current 422, or in addition to a counterweight current 422. In addition, the power counterweight 420 may also choose to shut down the operation of subsystem A 490, if its functionality is not required during the next clock cycle, by providing an inhibit signal on the inhibit burn signal line 492, thereby reducing the power consumption of the processor based system 400. Such inhibition of subsystem A 490 can be accomplished either in addition to generation of the



counterweight current 422, or as an alternative to the counterweight current 422 for the purpose of masking the power consumption of particular instructions that are executing (i.e., systems can be turned off to reduce current consumption in an amount proportional to a predicted value for a current clock cycle or randomly). One skilled in the art should appreciate that the discussion above related to subsystem A is exemplary only. In practice, the processor based system 400 will contain many subsystems (e.g., media processing unit, FPU, I/O circuits, random number generators, DRAM, etc.) that may be turned on even when they are not needed, to increase the power consumption of the processor based system 400, or turned off, to reduce the power consumption of the processor based system 400, as determined by the power counterweight 420. In one embodiment, located within the power counterweight 420 is a table (not shown) containing the on/off power requirements for each of the subsystems within the processor based system 400. Inhibit/burn signal lines 492 are connected to each of the subsystems so that they may be turned on or off as determined by the power counterweight 420. This will be particularly described below with reference to Figures 9 and 10.

[0055] To illustrate the effect the power predictor 418 and power counterweight 420 have on the total power consumed within the processor based system 400, attention is now directed at Figure 5. Figure 5 contains a table 520, similar to that shown in Figure 2, along with a power graph 530 illustrating power consumed by a processor that is executing the instructions in table 520.

[0056] More specifically, during clock cycle 1, instructions  $I_1$  thru  $I_4$  are found within pipeline stages W-R, respectively. Thus, instructions  $I_1$  thru  $I_4$  have a combined power consumption of 65. This is shown as current ( $I$ ) line 534. The power predictor 416 of Figure 4 provides the power estimate values for each of these instructions to the power correlator 418, which in turn, provides an arithmetic sum of these values to the power counterweight 420. The power counterweight 420 then produces a counterweight current 422 of 15 so that a maximum power threshold of 80 is consumed by the processor based system 400.

[0057] During clock cycle 2, the power predictors 416 provide the power estimate values for instructions  $I_2$  thru  $I_5$  to the power correlator 418 to produce a combined power value of 70. This value is provided to the power

counterweight 420. The power counterweight 420 therefore provides a counterweight current 422 with a value of 10 so that a maximum power threshold of 80 is consumed by the processor based system 400. Thus, during clock cycle 2, although the instructions that are executing are different than those executing in clock cycle 1, the total power consumed by the processor based system 400 is the same.

[0058] During clock cycle 3, the combined power estimate for instructions  $I_3$  thru  $I_6$  is 45. This value is provided to the power counterweight 420, to produce a counterweight current 422 of 35. Thus, the maximum power threshold that is consumed by the processor during clock cycle 3 is 80.

[0059] During clock cycle 4, the combined power estimate for instructions  $I_4$  thru  $I_7$  is 60. This value is provided to the power counterweight 420, to produce a counterweight current 422 of 20. Thus, the maximum power threshold that is consumed by the processor during clock cycle 4 is 80.

[0060] During clock cycle 5, the combined power estimate for instructions  $I_5$  thru  $I_8$  is 80. This value is provided to the power counterweight 420, to produce a counterweight current 422 of 0. Thus, the maximum power threshold that is consumed by the processor during clock cycle 5 is 80.

[0061] So, in graph 530, although we have different instructions proceeding thru the five pipeline stages during clock cycles 1-5, with different combined power consumption values (shown by line 534), the total power consumed remains at 80 (shown by line 532). This is because the power counterweight 420 produces a differential counterweight current 422, illustrated by shaded area 536.

[0062] Referring now to Figure 6, a block diagram is provided of a microprocessor 600, similar to that described above with reference to Figure 4. However, in this embodiment, rather than providing a power correlator 418 to arithmetically combine power profile values for each of the pipeline stages, a power counterweight 618 is added to each of the stages (r, a, m, and w) to produce distinct counterweight currents for each stage. Thus, the power estimate value is provided from the power predictor 616 in each stage directly to a stage specific power counterweight 618. The resulting maximum power consumed by the processor is identical to that shown in Figure 5, but without having to arithmetically combine the individual power estimate values.

[0063] Referring to Figure 7, and alternative embodiment of the present invention is shown within a microprocessor

700. One purpose of the invention described above has been to disassociate the power consumed by a microprocessor from the instructions it is executing. The embodiments shown in Figure's 4 and 6 disassociate the power by adding a counterweight current to obtain a maximum threshold value, thereby making the current drain invariant. In contrast, the embodiment of Figure 7 disassociates the power consumed by the microprocessor with instructions it is executing by adding a random counterweight current.

[0064] The processor 700 includes a fetch stage 702, a register stage 704, an ALU stage 706, a memory stage 708, and a write stage 710. Coupled to the fetch stage 702 (or alternatively the register stage 704) is a total power predictor 716. The total power predictor 716 is used to predict the total power that will be consumed during each clock cycle by instructions in the pipeline stages 704-710, similar to the power predictor 416 of Figure 4. Although the predictor 716 is not necessary to the embodiment shown in Figure 7, its prediction can be used to limit the random counterweight current 744 as will be further described below. Coupled to the total power predictor 716 is a counterweight generator 740 for producing a random

counterweight 744. The counterweight generator 740 is also connected to a random number generator 742.

[0065] Operationally, for each clock cycle of the microprocessor 700, the random number generator 742 provides a random value to the counterweight generator 740. The counterweight generator 740 then generates a random counterweight current 744 that is proportional to the random value provided by the random number generator 742. By adding the random counterweight current 744 to the power already consumed by the executing instructions, the total power consumed by the microprocessor 700 will be disassociated with the instructions it is executing, as will particularly illustrated below with reference to Figure 8. This has been accomplished without any estimation of the power that will be consumed, and without any coupling of the counterweight generator to the instructions that are executing. However, without the use of the total power predictor 716, it is possible that the power consumed by the microprocessor 700, during a particular clock cycle, may exceed an overall maximum threshold. So, the total power predictor 716 provides an estimate of the total power that will be consumed by the microprocessor 700 to the counterweight generator 740.

Within the counterweight generator 740 is a limiter that throttles the random counterweight current 744 to a predefined threshold. That is, whenever the random counterweight current 744, plus the total estimated power that is predicted by the total power predictor 716, exceeds a predefined threshold, the amount of the random counterweight current 744 is limited so that the total power consumed by the microprocessor 700 does not exceed the predefined threshold. One skilled in the art will appreciate that the predefined threshold may be set similar to that described above with reference to Figure 4, either at time of fabrication, or via software configuration. In addition, the random counterweight current 744 may be added to or subtracted from, a baseline power drain, on top of which the randomness is provided.

[0066] Referring now to Figure 8, a graph 800 is shown illustrating the power consumption of the microprocessor 700 utilizing the counterweight generator 740. For ease of illustration, the instructions executing during clock cycles 1-5 are the same as those illustrated in table 520 of Figure 5.

[0067] During the first clock cycle, instructions  $I_1$  thru  $I_4$  have a power consumption of 65. In addition, the random

number generator 742 provides a random value to the counterweight generator 740 to produce a random counterweight current 744 of 15. So, the total power consumed by the microprocessor is 80.

[0068] During the second clock cycle, instructions  $I_2$  thru  $I_5$  have a power consumption of 70 (5 more than in clock cycle 1). In addition, the random number generator 742 provides a random value to the counterweight generator 740 to produce a random counterweight current 744 of 5. So, the total power consumed by the microprocessor is 75. So, during clock cycle 2, even though the total power consumed by instructions  $I_2$  thru  $I_5$  is greater than the instructions executing during clock cycle 1, the total power consumed by the microprocessor 700 is 5 less than during clock cycle 1.

[0069] During the third clock cycle, instructions  $I_3$  thru  $I_6$  have a power consumption of 45. In addition, the random number generator 742 provides a random value to the counterweight generator 740 to produce a random counterweight current 744 of 40. So, the total power consumed by the microprocessor is 85. So, during clock cycle 3, even though the total power consumed by instructions  $I_3$  thru  $I_6$  is less than the instructions executing during clock cycles 1 or 2, the total power



consumed by the microprocessor 700 is greater than during those clock cycles.

[0070] During the fourth clock cycle, instructions I<sub>4</sub> thru I<sub>7</sub> have a power consumption of 60. In addition, the random number generator 742 provides a random value to the counterweight generator 740 to produce a random counterweight current 744 of 10. So, the total power consumed by the microprocessor is 70.

[0071] During the fifth clock cycle, instructions I<sub>8</sub> thru I<sub>8</sub> have a power consumption of 80. In addition, the random number generator 742 provides a random value to the counterweight generator 740 to produce a random counterweight current 744 of 15. However, in this instance, the counterweight current 744, plus the estimated power consumption of 80, exceeds a maximum power threshold of 85. So, the counterweight generator with limiter 740 limits the random counterweight current 744 to a value of 5, resulting in a total power consumption of 85.

[0072] What has been illustrated with respect to Figure's 3-6 are embodiments of the invention that disassociate power consumption of a processing system with the instructions it is executing by estimating the power that will be consumed during each clock cycle, and by

modulating the overall power consumption of a system by various means (adding counterweight currents, enabling/disabling subsystems, etc.) to mask the execution signature of the processing system, regardless of what instructions are in the pipeline. With respect to Figure's 7-8, an embodiment of the invention is described that disassociates the power consumption of a processing system with the instructions it is executing, by adding a randomized counterweight current to the total power consumed. Thus, whether the total power consumed is made invariant, or is randomized, there is no correlation between the total power consumed and the instructions that are being executed, that may be used by scientist to speculate or determine the instruction flow of a processing system incorporating the present invention.

[0073] The above-described embodiments disassociate the instantaneous power drain of the processor, either by randomizing the clock-by-clock power drain, or by maintaining a constant power drain. In most instances, this is performed by adding a counterweight current. It is also desirable, as mentioned above with reference to Figure 4, to turn on or shut down various subsystems on a microprocessor based system, if possible. By selectively

inhibiting power drain from selected subsystems, instantaneous power drain can be disassociated with instruction execution, while reducing the total power drain of the processor based system. This methodology will now be further described with reference to Figures 9-10, to which attention is now directed.

[0074] Referring to Figure 9, a microprocessor 900 is shown, similar to the one described above with respect to Figure 4. Like elements have like references with "9" replacing "4" in the hundreds digit. In addition, three subsystems 990, 991, and 993 identified as subsystems A, B, C are shown. As mentioned above with respect to Figure 4, these subsystems include FPU's, media processing unit units, or any other subsystem within a microprocessor than can be selectively turned on/off to increase or reduce the power drain of the microprocessor 900. The subsystems 990, 991, 993 are coupled to the power prediction system 920 via inhibit/burn signal lines 992. In Figure 9, the inhibit/burn signal lines 992 are shown as bi-directional because they are also intended to convey "subsystem available" information from each of the subsystems 991, 991, 991 to the power prediction system 920. That is, if the microprocessor 900 determines that any of its

subsystems will not be needed for processing, it will provide an indication to the power prediction system 920, on the signal lines 992, to indicate which, if any, of the subsystems may be inhibited, or powered off. The power prediction system can then choose to inhibit those available subsystems, as will now be further described.

[0075] Within the power prediction system 920 are subsystem power profiles 921 and subsystem inhibition/selection control 923. In one embodiment, the subsystem power profiles 921 provide an indication to the power prediction system 920 of the amount of power that each subsystem consumes when it is powered on. The power prediction system 920 utilizes the information contained in the subsystem power profiles 921 to select particular ones of the subsystems 990, 991, 993 to be powered down. The subsystem inhibition/selection control 923 works in conjunction with the power prediction system 920 to select particular ones of the subsystems 990, 991, 993 to be shut down as will now be described with reference to Figure 10.

[0076] Figure 10 provides a graph 1000 that illustrates 5 clock cycles of instruction execution within the microprocessor 900. Two power lines 1034, 1036 are shown. The first power line 1034 illustrates the power drain

within the microprocessor 900 during 5 clock cycles, without utilizing the power prediction system 920 of the present invention. The second power line 1036 illustrates the use of the subsystem inhibition/selection control 923, along with the power prediction system 920, to maintain a constant power drain for the microprocessor 900, thereby disassociating the instantaneous power drain from instruction execution.

[0077] During a first clock cycle, the power prediction system receives a power predictor 919 from the power correlator 918 and determines that the cumulative power drain in all stages of the microprocessor 900 will be less than a predetermined threshold. Rather than adding a counterweight current, the power prediction system 920 determines which of the subsystems 990, 991, 993 are available to be shut down, via the inhibit/burn signal lines 992, and communicates how much "counterweight" current is needed to maintain a constant power drain. The subsystem inhibition/selection control 923 utilizes the subsystem power profiles 921 to select an available subsystem to shut down. In this instance, subsystem A 990 provides the appropriate counterweight current, and is turned on, via inhibit/burn signal line 992.

[0078] During clock cycle 2, the power prediction system 920 determines that the predetermined power drain can be maintained by turning off subsystems A 990 and C 993, and therefore utilizes the subsystem inhibition/selection control 923 to turn off these subsystems.

[0079] During clock cycle 3, the power prediction system determines that to maintain a constant power drain, subsystems A 990, B 991, and C 993 must all be turned on, and therefore turns them on via inhibit/burn signal lines 992.

[0080] During clock cycle 4, the power prediction system determines that to maintain a constant power drain, subsystems A 990, and C 993 must be turned off, and therefore turns them off via inhibit/burn signal lines 992.

[0081] During clock cycle 5, the power prediction system determines that to maintain a constant power drain, subsystem B 991 must be turned off, and therefore turns it off via inhibit/burn signal lines 992.

[0082] At this point, one skilled in the art will appreciate that the microprocessor 900 has maintained a constant power drain during clock cycles 1-5 while draining less power than the embodiment described above with respect

to Figure 5. This is because it was possible to inhibit, or turn off, particular subsystems to maintain a constant power drain, rather than adding a differential power, or counterweight current.

[0083] Referring now to Figure 11, a graph 1100 is shown illustrating the use of the subsystem inhibition/selection control 923 of the present invention, in conjunction with the random counterweight circuitry described above with reference to Figures 7 and 8, to provide a randomized power drain shown by power line 1134, in response to an actual instruction power drain shown by power line 1132. That is, the random value generator 742 of Figure 7 provides a randomized desired power value, which in this instance may be positive or negative, to the subsystem inhibition/selection control 923. The inhibition/selection control 923 then turns on or turns off, available subsystems 990, 991, 993 to achieve a total power consumption that is random with respect to instruction execution.

[0084] While not specifically described above, one skilled in the art will appreciate that the ability to turn on or off particular subsystems within a microprocessor, to disassociate instantaneous power drain from the actual

power required to execute instructions, may be used as described in Figure 9, or may be combined with the counterweight current embodiment described above with respect to Figure 4, and/or may be combined with the random counterweight current embodiment described above with respect to Figure 7. That is, each embodiment has particular benefits, and may be utilized singly, or in combination with the other embodiments to obtain an optimum solution for a particular implementation. If a microprocessor has numerous subsystems that may be turned on/off, and if it is desired to reduce total power consumption to an absolute minimum, then the embodiment described with reference to Figure 9 may provide the optimum solution. However, if total power consumption is irrelevant, and disassociation of instruction execution with power consumption is the primary concern, then the embodiment described with reference to Figure 9 may be incorporated along with the counterweight currents of either Figure 4 and/or Figure 7.

[0085] Although the present invention and its objects, features, and advantages have been described in detail, other embodiments are encompassed by the invention as well. In addition to implementations of the invention using



hardware, the invention can be embodied in a computer usable (e.g., readable) medium configured to store a computer readable program code. The program code causes the enablement of the functions or fabrication, or both, of the invention disclosed herein. For example, this can be accomplished through the use of general programming languages (e.g., C, C++, etc.), hardware description languages (HDL) including Verilog HDL, VHDL, AHDL (Altera Hardware Description Language) and so on, or other programming and/or circuit (i.e., schematic) capture tools available in the art. The program code can be disposed in any known computer usable medium including semiconductor memory, magnetic disk, optical disc (e.g., CD-ROM, DVD-ROM, etc.) and as a computer data signal embodied in a computer usable (e.g., readable) transmission medium (e.g., carrier wave or any other medium including digital, optical or analog-based medium). As such, the code can be transmitted over communication networks including the Internet and intranets. It is understood that the functions accomplished by the invention as described above can be represented in a core (e.g., a microprocessor core) that is embodied in programming code and may be transformed to hardware as part of the production of integrated circuits.

Also, the invention may be embodied as a combination of hardware and software.

[0086] In addition, the present invention has been particularly characterized in terms of a CPU or microprocessor. In particular, the embodiments of the present invention described with reference to Figure's 3-11 portray its application within a 5-stage pipelined CPU based system 400. These specific embodiments and characterizations are presented herein as representative embodiments for the present invention. However, such description should by no means restrict application of the concept to a pipeline microprocessor, much less a 5-stage processor. Rather, the embodiments described that make the total power consumption of a processing system invariant, or random, may be applied to any processing system capable of executing instructions.

[0087] Furthermore, the discussion above has not attempted to describe the operation of the invention within the nuances of modern processor instruction flow. That is, one skilled in the art will appreciate that instruction flow within a pipeline microprocessor is often non-sequential, with flow being interrupted by branches, error conditions, etc. In addition, it is common for instruction

flow to stall or slip for a number of cycles, either when waiting on following pipeline stages to complete an operation, or when retrieving/storing data from/to memory. Such discussion has been removed so that the reader will more readily understand the invention. However, while not shown, it should be understood that the counterweight current of the present invention, particularly, the power counterweight block 420, shadows the pipeline of the processing system. That is, the counterweight block 420 is coupled to the interlock control of a processing system, and the power predictor 416 is coupled not only to the decoder, but also to the instruction pointer (and next instruction pointer) so that they may properly estimate and control the counterweight current in relation to the actual instruction flow of a modern processing system, regardless of slips or stalls in processing.

[0088] Those skilled in the art should appreciate that they can readily use the disclosed conception and specific embodiments as a basis for designing or modifying other structures for carrying out the same purposes of the present invention, and that various changes, substitutions and alterations can be made herein without departing from

the spirit and scope of the invention as defined by the appended claims.

[0089] What is claimed is:

to the extent that the same are not inconsistent with the prior art.